Puzzle

Kircho surprised Ivancho with interesting present. The present was puzzle consisted of 200 by 200 pixels and each pixel was made of one color by the model RGB.

The puzzle is divided in zones with Z*Z squares size. We can assume that we have new puzzle made of 200/Z rows and columns. Each zone is assumed as whole and you can't make operations in it.

All zones are part of original image, but at the moment are mixed in random order. The person who is solving it starts the game with removing one zone from it. He can't put it back or remove another zone during the game. After the player decides that this count of moves is enough, the player must return the removed zone in the empty place without rotating it in any way.

During the game the player has 4 possible moves. He can move the empty field in some of his neighboring fields as the zone in this field is moved in the empty place.

The player can make 4 types of moves. He can move the empty block in the one of the 4 neighboring zones. The field there comes on the empty place.

The four types of moves are described with capital English letters in this way:

- U – the zone over the empty filed is placed in its position and the the blank field is moved there.
- D – the zone under the empty filed is placed in its position and the the blank field is moved there.
- L - the zone left the empty filed is placed in its position and the the blank field is moved there.
- R – the zone right the empty filed is placed in its position and the the blank field is moved there.

For example if the empty field is situated in zone with coordinates (1,1) and the player makes move D, the zone which is in (2,1) will take position (1,1), and the empty field - (2,1).

If the empty zone is in on the border of the puzzle some moves can't be made. For example if the empty zone is in position (0,0), you can't make U or L move.

The main goal of the puzzle is to make the messed up picture as close as you can to the original one. Of course the best you can do is to make the both pictures equal.

For each move you are given 1 penalty point. The removing of the zone and returning it back to the puzzle doesn't cost points.

After the game has finished the difference between the original and the messed up picture is calculated in this way. For each pixel of the puzzle (200*200) is calculated the difference between the messed picture and the original one for each pixel. The difference between two images is calculated by the formula:

$$\sum_{i=0,j=0}^{i<200,j<200} \sqrt{(Rij.new - Rji.old)^2 + (Gij.new - Gij.old)^2 + (Bij.new - Bij.old)^2}$$

For each i,j where $\sqrt{(Rij.new - Rji.old)^2 + (Gij.new - Gij.old)^2 + (Bij.new - Bij.old)^2}$ is not integer, it is rounded to the closest integer less than it. (floor)

Where $Rij.new$ is the value of red in the pixel with coordinates i,j of the new image, $Rji.old$ - is the value of red in the pixel with coordinates i,j of the original image and so on.

Ivancho lost interest in this puzzle because it is too difficult for him. You don't think so. You decide to write program **puzzle**, which solves such puzzles.

Puzzle

Input

On the first row of the input file `puzzle.in` is given the number Z - the size of the zones.

On the next 200 rows are given 600 numbers, separated by interval, representing the mixed image. For each pixel is entered one triple of numbers. The first number in it represents the red in the pixel, the second one – the green and the las one – the blue.

The next 200 rows consist again of 600 numbers representing the original image.

Output

On the first row of output file `puzzle.out` you have to write two numbers – the coordinates of the zone you want to remove. The first number is its row and the second number is its column.

Note: The zones are numerated from (0,0) to (200/Z -1, 200/Z -1).

On the second row you have to print string containing of symbols 'U','D','L','R', which represent your moves. The first symbol represents the first move and so on.

Constrains

```
N = 200
Z = 1;2;5;10 или 20
0 <=  the value of red green and blue in each pixel <= 255
```

Time limit: 10 сек.

Memory limit: 256MB

Note: You have limit of 1 000 000 moves of type U,L,R,D

Evaluation

In standings, each solution receives $\left(\frac{minScore+1}{yoursScore+1}\right)^2$ percent of the points for the test. minScore is defined as the minimal count of penalty points that someone has received for that test and yoursScore is the number of penalty points your solution for that test has.

If in some point of the puzzle solving your moves set the blank zone out of the puzzle you will get no points for that test case.

If you are trying to remove invalid zone from the puzzle at the beginning you will receive zero points for this test.

If in your output there are symbols that doesn't mean valid move or removed zone you will receive zero points.

If your program makes more than 1 000 000 moves on one test it will get 0 points for it.

Tests

In 50% from the test it is guaranteed that solution exists.

Example

You can download the example input file and the output file uploaded in the CodeIT website.

Visualizer

The jury has prepared special visualizer for the task. You can find how to use it in the file **instructions**, published on the task page on CodeIT website.